UNITED STATES PATENT AND TRADEMARK OFFICE

---------------------------------

BEFORE THE BOARD OF PATENT APPEALS
AND INTERFERENCES

---------------------------------

*Ex parte* Dennis Steven DeLorme, Margaret Rose Fenlon, Robert Thaddeus Gintowt,
Alan Leon Levering, Thomas Marcus McBride, Michael John Oberholtzer, Jeffrey John Parker,
Richard Michael Theis, Jerry Leroy Von Berge

---------------------------------

Appeal No.  _____
Application No. 10/777,869

---------------------------------

APPEAL BRIEF

---------------------------------

## IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant:          Dennis Steven DeLorme et al.   Art Unit:    2161
Application No.:  10/777,869                      Examiner:   Paul Kim
Filed:              February 12, 2004
For:                METHOD OF CONVERTING A FILESYSTEM WHILE THE
FILESYSTEM REMAINS IN AN ACTIVE STATE

---

Mail Stop Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

## APPEAL BRIEF

### I.  REAL PARTY IN INTEREST

This application is assigned to International Business Machines Corporation, of Armonk,

New York.

### II.  RELATED APPEALS AND INTERFERENCES

There are no related appeals or interferences.

### III.  STATUS OF CLAIMS

Claims 1-35 are pending in the Application, stand rejected, and are now on appeal.

### IV.  STATUS OF AMENDMENTS

There have been no amendments filed subsequent to the final rejection mailed

February 5, 2007.

## V. SUMMARY OF CLAIMED SUBJECT MATTER

Applicants' invention is generally directed to a filesystem conversion process that does not require shutting down a filesystem to perform the conversion, ensures all objects within the filesystem are converted, and, from the perspective of a user, does not impact the performance and operation of the file system (Application, page 5, lines 2-5).

Filesystem conversion processes conventionally have been used to update filesystems from one type to another, e.g., to upgrade filesystems to newer, better performing, and more feature rich versions. Conventional conversion processes have often required filesystems to be shut down and restarted in order to complete a filesystem conversion. Doing so, however, makes the filesystem unavailable for productive use, and in many applications, such as high availability and mission critical applications, shutting down a filesystem for any amount of time is highly undesirable. (Application, page 2, line 17 to page 3, line 13).

Embodiments consistent with the invention, on the other hand, address the drawbacks of conventional filesystem conversion processes by enabling a filesystem to be converted while maintained in an active state. Independent claims 1 and 24, for example, respectively recite a method (Application, page 13, lines 12-14) and a program product (Application, page 9, line 16 to page 10, line 11) for converting a filesystem from a first type to a second type. (Application, page 5, lines 6-11). To perform the conversion, a list is created of directories of the first type in the filesystem to convert (Application, page 16, lines 3-4, page 16, line 10 to page 17, line 8, Fig. 4, block 402, Fig. 5, blocks 502-510). Then, each directory in the list is converted to the second type while maintaining the filesystem in an active state (Application, page 16, lines 1-9, page 17, line 21 to page 21, line 15, Fig. 3, blocks 306-308, Fig. 7, blocks 702-734).

Independent claims 20, 25 and 26 respectively recite a method (Application, page 13, lines 12-14), program product (Application, page 9, line 16 to page 10, line 11) and apparatus (Application, page 7, lines 9-19, Fig. 1, block 10) for converting a filesystem from a first type to a second type. (Application, page 5, lines 6-11). To perform the conversion, a conversion process is executed to convert each directory of the first type in the filesystem into the second type while maintaining the file system in an active state. (Application, page 16, lines 1-9, page 17, line 21 to page 21, line 15, Fig. 3, blocks 306-308, Fig. 7, blocks 702-734) Then, the

conversion process is terminated when every directory of the first type in the filesystem has been converted to the second type. (Application, page 21, lines 13-15, Fig. 7, block 734).

For program product claims 24 and 25, support for the recited program code and signal bearing medium can be found at page 9, line 16 to page 10, line 11 of the Application. For apparatus claim 26, support for the claimed processor and memory can be found at page 7, lines 20-21 and Fig. 1, blocks 12 and 14 of the Application.

Other support for the claimed subject matter may generally be found in Figs. 3-5 and 7-8 and the accompanying text at pages 13-24 of the Application as filed. In addition, it should be noted that, as none of the claims recite any means plus function or step plus function elements, no identification of such elements is required pursuant to 37 CFR §41.37(c)(1)(v). Furthermore, there is no requirement in 37 CFR §41.37(c)(1)(v) to provide support for the subject matter in the separately argued dependent claims, and so no discussion of any of these claims is provided.

## VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

A.  Claims 1-2, 5, 7-9, 20-22, 24-26 and 28 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over U.S. Patent No. 6,571,231 to Sedlar (*Sedlar*) in view of U.S. Patent No. 5,625,804 to Cooper (*Cooper*).

B.  Claims 3-4, 23 and 34 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over *Sedlar* in view of *Cooper* and further in view of Official Notice (*Official Notice*).

C.  Claim 6 stands rejected under 35 U.S.C. § 103(a) as being unpatentable over *Sedlar* in view of *Cooper* and further in view of U.S. Patent No. 5,873,097 to Harris et al. (*Harris*).

D.  Claims 10-11, 18-19 and 30 stand rejected under 35. U.S.C. § 103(a) as being unpatentable over *Sedlar* in view of *Cooper* and further in view of West et al. (NPL, "Batch Processing" excerpt fro "Sams Teach Yourself Macromedia Fireworks MX in 24 Hours"), published on 4 December 2002 (*West*), and *Harris*.

E.  Claims 14 and 32 stand rejected under 35. U.S.C. § 103(a) as being unpatentable over *Sedlar* in view of *Cooper* and further in view of U.S. Patent Application Publication No. 2003/0217057 by Kuroiwa et al. (*Kuroiwa*).

F.  Claims 12, 15-17, 29 and 31 stand rejected under 35. U.S.C. § 103(a) as being unpatentable over *Sedlar* in view of *Cooper* and further in view of U.S. Patent No. 6,728,907 to Wang et al. (*Wang*).

G.  Claim 13 stands rejected under 35. U.S.C. § 103(a) as being unpatentable over *Sedlar* in view of *Cooper* and *Wang*, and further in view of *Official Notice*.

H.  Claim 27 stands rejected under 35. U.S.C. § 103(a) as being unpatentable over *Sedlar* in view of *Cooper* and further in view of Dubinski (NPL, "Non-recursive tree walks"), by John Dubinski, published on 1 May 1996 (*Dubinski*).

I.  Claims 33 and 35 stand rejected under 35. U.S.C. § 103(a) as being unpatentable over *Sedlar* in view of *Cooper* and further in view of U.S. Patent No. 6,338,072 to Durand et al. (*Durand*).

## VII.  ARGUMENT

Applicants respectfully submit that the Examiner's rejections of claims 1-35 are not supported on the record, and should be reversed. All such claims have been rejected as being obvious over the prior art cited by the Examiner. A *prima facie* showing of obviousness, however, requires that the Examiner establish that the differences between a claimed invention and the prior art "are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art." 35 U.S.C. §103(a). Such a showing requires that all claimed features be disclosed or suggested by the prior art. Four factors generally control an obviousness inquiry: 1) "the scope and content of the prior art"; 2) the "differences between the prior art and the claims"; 3) "the level of ordinary skill in the pertinent art"; and 4) objective evidence of nonobviousness. KSR International Co. v. Teleflex Inc., 127 S. Ct. 1727, 1734 (2007) (quoting Graham v. John Deere Co. of Kansas City, 383 U.S. 1, 17-18 (1966)). Furthermore, while there is no rigid requirement for an explicit "teaching, suggestion or motivation" to combine references, there must be some evidence of "a reason that would have prompted a person of ordinary skill in the relevant field to combine the elements in the way the claimed new invention does" in an obviousness determination. KSR, 127 S. Ct. at 1731.

Applicants respectfully submit that, in the instant case, the Examiner has failed to establish a *prima facie* case of obviousness as to claims 1-35, as such, the rejections thereof should be reversed. Applicants' remarks in rebuttal to the Examiner's rejections are presented

below, starting with the relevant independent claims and followed up by a discussion of selected dependent claims. In some cases, specific discussions of particular claims are not made in the interests of streamlining the appeal. The omission of a discussion with respect to any particular claim, however, should not be interpreted as an acquiescence as to the merits of the Examiner's rejection of the claim, particularly with respect to claims reciting features that are addressed in connection with the rejections applied to other claims pending in the appeal.

A.    Claims 1-2, 5, 7-9, 20-22, 24-26 and 28 are non-obvious over *Sedlar* and *Cooper*.

Independent Claim 1

Claim 1 generally recites a method for converting a filesystem from a first type to a second type. The method comprises the steps of: generating a list of directories of the first type in the filesystem to convert, and converting each directory in the list to the second type while maintaining the file system in an active state.

In rejecting claim 1, the Examiner relies on the combination of *Sedlar* and *Cooper*. The Examiner asserts that *Sedlar* discloses the generation of a list of directories, specifically at col. 3, lines 50-61 and in Figs. 1-3, which refer to a directory links table. It is important to note, however, that claim 1 recites "generating a list of directories . . . to convert." Put another way, the claimed list is not just a list of directories, it is a list of directories "to convert."

The cited passages in *Sedlar*, in particular Fig. 3 and the passage at col. 3, lines 50-61, merely disclose a directory links table. The simple fact that this table includes directories, however, falls short of disclosing "a list of directories . . . to convert." Indeed, *Sedlar* does not disclose or suggest converting a filesystem from a first type to a second type, so *Sedlar* does not disclose or suggest that the disclosed table is or could be used as a list of directories to be converted to another type.

In addition, as the Examiner has implicitly acknowledged, *Sedlar* does not disclose converting a filesystem from a first type to ta second type, or converting each directory in the list to a second type while maintaining a filesystem in an active state. Indeed, *Sedlar* is completely silent with respect to the concept of filesystem conversion.

*Cooper*, however, fails to address the shortcomings of *Sedlar*. The Examiner relies on col. 2, lines 44-59 of *Cooper* for allegedly disclosing converting directories in a list to a second type while maintaining a filesystem in an active state. *Cooper*, however, discloses the conversion of individual records to different formats while maintaining a multiprocessor system in an active state. *Cooper* does not disclose that the records are directories, files, and in fact, does not mention "directories", "files", "folders", or any other analogous structures related to filesystem. Moreover, *Cooper* does not disclose maintaining a <u>filesystem</u> in an active state while changing directories in that filesystem to different types.

In addition, the conversion of records in *Cooper* does not include the generation of any list of records to convert. Instead, records are converted in *Cooper* as they are accessed by individual processors (*see, e.g.,* Figs. 2 and 4a). *Cooper* does not generate any list of records to be converted, and in fact, does not retrieve records to be converted from a list or otherwise rely on any type of list in connection with converting records. Furthermore, *Cooper* does not even disclose that the individual records being converted are linked to one another as would be found in a list of directories in a filesystem.

Claim 1 recites converting "each directory in the list," and as such, the fact that *Cooper* discloses converting the format of individual records falls short of disclosing or suggesting the conversion of directories that are identified in a "list of directories . . . to convert," as required by claim 1.

As such, the combination of *Sedlar* and *Cooper* falls short of disclosing each and every feature of claim 1. In particular, neither reference discloses or suggests the generation of a list of directories (or any other data structures) "to convert", and neither reference discloses or suggests converting directories (or any other data structures) "in [a] list [to convert]" to a different type.

In addition, neither reference discloses or suggests converting a filesystem to a different type while maintaining that filesystem in an active state. *Sedlar* does not address this particular feature, and maintaining a multiprocessor system in an active state while converting individual records to a different format (as disclosed in *Cooper*), does not specifically disclose or suggest the conversion of a <u>filesystem</u> to a different type while that filesystem is in an active state.

Applicants also respectfully submit that the rejection is improperly reliant on hindsight. Specifically, neither reference discloses or suggests converting a filesystem to a different type, much less doing so while maintaining that filesystem in an active state. *Sedlar* merely discloses a filesystem, but no manipulations analogous to converting the filesystem are disclosed or suggested by the reference. *Cooper* does not even disclose filesystems, directories, or any analogous concepts, and the fact that the reference merely discloses converting individual records to different formats falls short of disclosing or suggesting the conversion of a <u>filesystem</u>. Applicants therefore submit that *Cooper* does not provide any motivation to one of ordinary skill in the art to modify *Sedlar* to implement a runtime filesystem conversion process. Applicants also submit that Examiner has not otherwise provided any credible reason why one of ordinary skill in the art would be motivated to modify *Sedlar* to implement a runtime filesystem conversion process. Absent any evidence of such a motivation, or even any professed reason to make a modification, the rejection is necessarily reliant on hindsight, and as such, the rejection cannot be maintained.

The Examiner attempted to rebut Applicants' arguments on pages 15-17 of the Final Office Action. First, the Examiner argued that Applicants were attempting to attack the references individually in asserting that *Sedlar* did not disclose converting a filesystem from a first type to a second type, and that *Sedlar* did not disclose that a table could be used as a list of directories to be converted to another type. However, Applicants' arguments in this regard were in a larger context of explaining why <u>neither</u> reference disclosed the claimed feature, and indeed, Applicants later explained in the Amendment and Response why *Cooper* likewise did not disclose the claimed feature. Claim 1 recites "generating a list of directories . . . to convert," and it is indisputable that neither reference discloses the generation of a list of directories to be converted. As Applicants noted above, *Sedlar* does not disclose the generation of a list of directories to be converted, nor does *Cooper*. It is therefore logical to assume that the combination of these references likewise does not disclose or suggest this feature of claim 1. Given that the combination must disclose or suggest each and every feature of the claim, no *prima facie* case of obviousness can be established where, as here, the combination does not disclose or suggest a feature of the claim.

Second, the Examiner argued that Applicants were also attempting to attack the references individually in asserting that *Sedlar* did not disclose converting a filesystem from a first type to a second type, and that *Sedlar* did not disclose converting each directory in the list to a second type while maintaining the filesystem in an active state. Again, Applicants' arguments in this regard were in a larger context of explaining why <u>neither</u> reference disclosed the claimed feature, and indeed, Applicants later explained in the Amendment and Response why *Cooper* likewise did not disclose the claimed feature. Claim 1 recites "converting each directory . . . while maintaining the filesystem in an active state," and it is indisputable that neither reference discloses the conversion of a <u>directory</u> while maintaining a <u>filesystem</u> in an active state. *Sedlar* only discloses the concepts of directories and filesystems, but doesn't disclose converting a filesystem, while *Cooper* discloses the conversion of data in the abstract, but lacks any disclosure or suggestion that the data being converted could be a directory or other part of a filesystem, and that such conversion could be done while a filesystem in particular was maintained in an active state. Given that the combination must disclose or suggest each and every feature of the claim, no *prima facie* case of obviousness can be established where, as here, the combination does not disclose or suggest a feature of the claim.

Third, the Examiner argued that Applicants are arguing non-claimed features by arguing that *Cooper* does not disclose that the records being converted in *Cooper* could be directories or files. The Examiner then argued that one of ordinary skill in the art could interpret the limitation "converting each directory in the list to the second type" to mean the conversion of all data, records and/or files within the directory to another format. It is completely nonsensical that the Examiner could interpret the limitation "converting [a] directory" to mean "converting everything in a directory except the directory itself," as the Examiner apparently does here, since *Cooper* does not disclose or suggest converting any data structure that conforms to a directory in a filesystem. The fact that the data structures being in converted in *Cooper* might be stored in directories, as is apparently the Examiner's position, falls far short of converting any directories within which such data structures might be stored. Claim 1 recites "converting [a] directory," not "converting [the contents of a] directory," and it requires a rather tortured reading of the claim to come to any other conclusion. *Cooper* simply does not disclose converting any directory in a

filesystem, so the Examiner's attempt to circumvent this shortcoming of *Cooper* by effectively reading additional language into the claim is improper. Certainly, claim 1 supports the ability to convert the contents of a directory along with the directory itself; however, the plain language of the claim still requires that the directory be converted itself, otherwise the term is read out of the claim entirely.

Fourth, the Examiner argues that *Cooper*'s disclosure of converting data "from one format to another while maintaining system operations" is sufficient to disclose maintaining a filesystem in an active state while changing directories in the filesystem to different types. The only way this disclosure could be sufficient is if the concepts of filesystems and converting directories were read out of the claim entirely. The Examiner has improperly placed no weight on the precise language of the claim that focuses the conversion on directories in a filesystem.

It is evident from a number of the Examiner's arguments that hindsight is improperly being used in formulating the rejection. The cited references establish nothing more than the conventionality of basic concepts like directories in filesystems, lists of directories, and the conversion of data while a computer system is active. There is nothing in either reference beyond a general correspondence with items from this laundry list of concepts. Neither reference even addresses the concept of converting a filesystem while that filesystem is maintained in an active state, much less doing so by generating a list of directories from the filesystem that require conversion, and then converting directories in that list. The Examiner has essentially partitioned the language in claim 1 to such an extent that the overall concepts recited in the claim are effectively read out of the claim. As but one example, by finding a "list of directories" in *Sedlar* and the general conversion of data in *Cooper*, the Examiner effectively reads out of the claim the fact that the claim actually recites not just any list of directories, but a list of "directories . . . to convert." This type of analysis is precisely the type of hindsight-based reasoning that should not be condoned within the context of patentability analysis, as practically any combination-type invention can ultimately be broken down into purely conventional elements, and without any evidence of some sort of reasoning or justification as to why one of ordinary skill in the art would make the combination, very few inventions would ever be able to overcome such a difficult hurdle to patentability.

Applicants therefore respectfully submit that claim 1 is non-obvious over the combination of *Sedlar* and *Cooper*. Reversal of the Examiner's rejection, and allowance of claim 1, and of claims 2-19 that depend therefrom, are therefore respectfully requested.

## Independent Claim 20

Claim 20 generally recites a method for converting a filesystem from a first type to a second type. The method comprises the steps of: executing a conversion process to convert each directory of the first type in the filesystem into the second type while maintaining the filesystem in an active state, and terminating the conversion process when every directory of the first type in the filesystem has been converted to the second type.

In rejecting claim 20, the Examiner again relies on *Sedlar* and *Cooper*. However, as discussed above in connection with claim 1, the combination of *Sedlar* and *Cooper* does not disclose or suggest a method of converting a filesystem from a first type to a second type while maintaining the filesystem in an active state. *Sedlar* discloses a filesystem, but does not disclose or suggest converting the filesystem to another type, much less doing so while maintaining that filesystem in an active state. Likewise, *Cooper*, while disclosing converting data records to different formats while maintaining a multiprocessor system in an active state, does not even mention filesystems, and certainly does not suggest that the data record conversion process disclosed therein can be used to convert a filesystem.

Applicants respectfully submit that the rejection of claim 20 is improperly reliant on hindsight. In order to establish a *prima facie* case of obviousness for claim 20, the Examiner must present some objective evidence of a motivation in the art to combine *Sedlar* and *Cooper* to convert a filesystem from one type to another while maintaining that filesystem in an active state. *Sedlar* itself does not provide any such motivation, given that there is no discussion in the reference directed to converting a filesystem to a different type. *Cooper*, likewise, fails to provide any such motivation, given that the reference does not discuss filesystems, or otherwise suggest that the data record conversion process disclosed in the reference could be used in converting a filesystem. Neither reference even appreciates the desirability of converting a filesystem while maintaining that filesystem in an active state. In addition, the Examiner has

cited no other reference providing any of the motivation lacking in *Sedlar* and *Cooper*. Applicants submit that the Examiner has failed to provide any objective reason why one of ordinary skill in the art would be motivated to combine *Sedlar* and *Cooper* to convert a filesystem while maintaining that filesystem in an active state.

Applicants submit that in this case, Applicants' disclosure has simply been used as a blueprint, given that *Sedlar* and *Cooper* otherwise have no relationship to one another in terms of the problems addressed thereby or the solutions used to address such problems. Neither reference even addresses the same problem as that faced by Applicants, and Applicants therefore respectfully submit that no *prima facie* case of obviousness has been established for claim 20.

The Examiner also attempted to rebut Applicants' arguments as to claim 20 on page 18 of the Final Office Action. The Examiner in particular argued that one of ordinary skill in the art would be motivated to combine the references such that the "directory links table" in *Sedlar* could be used with *Cooper*'s data conversion technique to convert all of the records or files in a directory to another format. The Examiner's statement in this regard is entirely conclusory in nature, as there is nothing in either reference that supports any argument that this motivation would be apparent to one of ordinary skill in the art. There must be some objective reason to make the combination, and the Examiner has established that no such reason exists.

In addition, the Examiner basis for the combination is once again based upon a faulty reading of the language in the claim. The Examiner has apparently taken the position that "converting a directory" means converting the contents of a directory instead of the directory itself. This rhetorical slight of hand, however, disregards the plain language of the claim, and is entirely inconsistent with the claims and the file history. While a filesystem conversion process consistent with the invention may require the conversion of objects such as files and directories that are stored within directories, the claimed filesystem conversion process also requires, by the plain language of the claim, that directories themselves be converted. Any other reading is inconsistent with the language of the claim, and thus an improper analysis of the claim.

As such, Applicants submit that the Examiner has failed to establish a *prima facie* case of obviousness as to claim 20. Reversal of the Examiner's rejection and allowance of claim 20, and of claims 21-23 which depend therefrom, are therefore respectfully requested.

## Independent Claim 24

Claim 24 recites in part program code configured to generate a list of directories of a first type in a filesystem to convert, and convert each directory in the list to a second type while maintaining the filesystem in an active state. As discussed above in connection with claim 1, this combination of features is not disclosed or suggested by the combination of *Sedlar* and *Cooper*. Accordingly, claim 24 is non-obvious over these references for the same reasons as presented above for claim 1. Reversal of the Examiner's rejections, and allowance of claim 24 are therefore respectfully requested.

## Independent Claims 25 and 26

Claims 25 and 26 recite in part program code configured to initiate a conversion process to convert each directory of a first type in a filesystem into a second type while maintaining the filesystem in an active state, and terminate the conversion process when every directory of the first type in the filesystem has been converted to the second type. As discussed above in connection with claim 20, this combination of features is not disclosed or suggested by the combination of *Sedlar* and *Cooper*. Accordingly, claims 25 and 26 are non-obvious over these references for the same reasons as presented above for claim 20. Reversal of the Examiner's rejection and allowance of claims 25-26, and of claims 27-35 which depend therefrom, are therefore respectfully requested.

## Dependent Claims 2, 5 and 7-9

Claims 2, 5 are not argued separately.

## Dependent Claim 7

Claim 7 depends from claim 1, and additionally recites that the step of generating further includes the steps of:

   a) adding a root directory as a current entry in the list;

   b) identifying a child directory of the current entry in the list;

   c) appending the identified child directory to the list;

d)  repeating steps b) and c) for each child directory within the current entry;

e)  changing a next directory in the list immediately following the current entry to be the current entry, if the next directory exists in the list; and

f)  repeating steps b) - e) until no next directory exists in the list.

In rejecting claim 7, the Examiner argues that *Sedlar* discloses each of these concepts at Figs. 1-3 and Col. 3, lines 50-61. The cited passages, however, merely disclose a directory links table including a list of directories and the links therebetween. There is nothing in the cited passages that describes or suggests any of the specific steps recited in claim 7, e.g., "adding a root directory," "identifying a child directory," "appending the child directory to the list," and "changing a next directory in the list." The Examiner's analysis of claim 7 is superficial at best, and appears to rely only on the fact that elements such as entries, lists, and directories are disclosed in the cited passages. There is no disclosure of how these elements are processed in the precise manner recited in claim 7. As such, the Examiner has failed to meet the burden required to establish a *prima facie* case of obviousness as to claim 7. Reversal of the Examiner's rejection of claim 7 is therefore respectfully requested.

## Dependent Claims 8-9 and 21

Claims 8-9 and 21 are not argued separately.

## Dependent Claim 22

Claim 22 depends from claim 20, and additionally recites the step of generating a list of directories of the first type in the filesystem to convert. In rejecting claim 22, the Examiner relies on *Sedlar*, col. 3, lines 50-63 and Figs. 1-3 for allegedly disclosing generating a list of directories. However, as Applicants discussed above in connection with claim 1, *Sedlar* does not disclose or suggest generating a list of "directories . . . to convert." The cited passages are completely silent with respect to either how the list is generated, or that such a list could be generated for the purpose of converting the contents of that list to a different format. Applicants

are not claiming lists of directories in the abstract. Instead, claim 22 is focused upon generating a list that contains directories from a file system to be converted by a filesystem conversion process. *Sedlar*, having no appreciation for filesystem conversion, does not disclose or suggest any such type of list. In fact, given that the claimed list contains those directories "to convert" during a filesystem conversion process, that list will change as directories are removed after they are successfully converted. Neither *Sedlar* nor *Cooper* discloses or suggests any analogous functionality for maintaining a list of directories still awaiting conversion. Accordingly, no *prima facie* case of obviousness has been established for claim 22, and claim 22 is therefore non-obvious over the references cited by the Examiner. Reversal of the Examiner's rejection of claim 22 is therefore respectfully requested.

### Dependent Claim 28

Dependent claim 28 is not argued separately.

**B.**     Claims 3-4, 23 and 34 are non-obvious over *Sedlar, Cooper* and *Official Notice*.

### Dependent Claims 3 and 23

Claim 3 depends from claim 1 additionally recites that the step of converting further includes the steps of: retrieving an identifier of a directory in the list, converting the directory to a second-type directory, and activating the second-type directory. Claim 23 depends from claim 20 and recites similar subject matter. In rejecting these claims, the Examiner adds *Official Notice* to *Sedlar* and *Cooper.* The Examiner simply disregards the fact that neither reference discloses activating a directory with the conclusory allegation that it would have been obvious to activate a directory. Such a conclusory observation, however, absent any supporting evidence, falls far short of raising to the level necessary to sustain a *prima facie* case of obviousness. Reversal of the Examiner's rejections of claims 3 and 23 is therefore respectfully requested.

### Dependent Claim 4

Claim 4 is not argued separately.

Dependent Claim 34

Claim 34 depends from claim 26 and additionally recites that the program code is further configured to, for a particular directory already converted, convert the particular directory back to the first type. In rejecting this claim, the Examiner adds *Official Notice* to *Sedlar* and *Cooper*, arguing that it would have been obvious to revert a directory already converted back to its original type. Once again, the Examiner's reliance on *Official Notice* has no support on the record, and is simply a conclusory argument used only because *Sedlar* and *Cooper* are admittedly silent on the concept. *Cooper*, in particular, does not disclose any conversion back to an original format. Applicants submit that, in the least, the Examiner is required to provide some evidence that one of ordinary skill in the art would in fact be motivated to convert a directory back to its original type after it has already been converted. The Examiner cannot meet this burden in this instance, and as such, the Examiner has failed to meet the burden required to establish a *prima facie* case of obviousness. Reversal of the Examiner's rejection of claim 34 is therefore respectfully requested.

C.      Claim 6 is non-obvious over *Sedlar, Cooper* and *Harris*.

Claim 6 depends from claim 1 and therefore incorporates all of the subject matter thereof. *Harris* does not and has not been asserted by the Examiner to disclose or suggest the underlying features recited in claim 1. Accordingly, irrespective of whether or not *Harris* discloses or suggests any of the additional features of claim 6, claim 6 is patentable over the cited references for the same reasons as claim 1. Reversal of the Examiner's rejection, and allowance of claim 6, are therefore respectfully requested.

D.      Claims 10-11, 18-19 and 30 are non-obvious over *Sedlar, Cooper, West and Harris*.

Dependent Claim 10

Claim 10 depends from claim 1 and additionally recites that the step of converting further includes the steps of: a) creating a second-type root directory, b) creating a second-type directory corresponding to a particular directory in the list, c) generating a respective link in the second-

type directory for each child object of the particular directory, d) activating the second-type directory, and e) removing the particular directory from the list.

In rejecting the claim, the Examiner adds *West* to the rejection, arguing that the reference discloses creating the second-type root directory, creating the second-type directory corresponding to the particular directory in the list, and generating a link in the second-type directory for each child object of the particular directory. In making this argument, the Examiner relies on Figs. 18.3 and 18.9 and pp. 1 and 6-7 of *West*. The cited passages, however, merely disclose the creation of directories for entirely different purposes, namely in connection with batch processing of images. The citation of *West* at all is questionable, much less the citation of the reference for the particular claim limitations at issue with claim 10. The Examiner has failed to provide any analysis as to how image processing techniques are relevant to the claimed filesystem conversion process, and in particular, how these techniques are related to the aforementioned steps, which are performed specifically in connection with "converting each directory in [a] list to [a] second type while maintaining [a] filesystem in an active state" (since these steps are recited as being part of the "converting step").

Furthermore, the Examiner relies on *Harris* for allegedly disclosing the activation of a directory and the removal of a directory from the list. Once again, however, the Examiner is improperly relying on hindsight, as the Examiner is merely taking isolated passages from the reference without a consideration of whether there is any reason or motivation for one of ordinary skill in the art to do incorporate those isolated teachings into the proposed combination. The Examiner has not and cannot show any reason that one of ordinary skill in the art would look to *West* and *Harris* to convert a directory to another format using the specific steps recited in claim 10. The Examiner is using Applicants' specification as a blueprint with which to construct the rejection, without any consideration whatsoever into what the references teach as a whole. For this reason, the Examiner's rejection is in error and should be reversed.

Dependent Claim 11

Claim 11 depends from claim 10, and additionally recites the step of creating a data structure associated with the second-type directory, the data structure including a first anchor

point that is associated with a parent directory of the directory and a second anchor point associated with a parent directory of the second-type directory.

In rejecting claim 11, the Examiner simply makes a conclusory statement that it would have been obvious to combine the cited references. Under no circumstance should the Examiner's stated basis for this rejection ever be sufficient to meet the burden required to establish a *prima facie* case of obviousness. The cited passage at col. 5, lines 19-49 of *Harris* is completely irrelevant to the concept of providing two anchor points in a data structure for a directory, and the Examiner provides no analysis of the reference or any explanation why the cited language in the reference applies to the language in claim 11. Reversal of the Examiner's rejection of claim 11 is therefore respectfully requested.

## Dependent Claim 18

Claim 18 depends from claim 1 and additionally recites that the step of activating further includes the steps of identifying a data structure associated with the directory, changing the data structure to be associated with the second-type directory, and removing the directory. The Examiner cites *Harris*, col. 5, lines 19-49, but as with claim 11, the passage is of only tangential interest with respect to the claim language, and the Examiner has provided no argument beyond a simple conclusory statement. Applicants submit that the Examiner has not met the burden required to establish a *prima facie* case of obviousness as to claim 18. Reversal of the Examiner's rejection is therefore respectfully requested.

## Dependent Claims 19 and 30

Claims 19 and 30 are not argued separately.

E.     Claims 14 and 32 are non-obvious over *Sedlar, Cooper* and *Kuroiwa*.

## Dependent Claim 14

Claim 14 depends from claim 1 and additionally recites determining a usage rate of a particular directory before converting that directory, and postponing converting the particular

directory based on the usage rate. Claim 32 depends from claim 26 and recites similar subject matter. In rejecting these claims, the Examiner adds *Kuroiwa* to *Sedlar* and *Cooper*. The cited passages of *Kuroiwa*, however, merely disclose interrupting a conversion process for a content using method based upon CPU load. The passages deal with conversion of media content, and are entirely irrelevant to the concept of determining a usage rate of a directory to be converted when converting a filesystem. The rejection is unfortunately emblematic of the superficial nature of most of the rejections in the Final Office Action, as there is no evidence presented whatsoever as to how the cited passages would motivate one of ordinary skill in the art to modify *Sedlar* and *Cooper* to arrive at Applicants' claimed invention. The Examiner has again fallen far short of meeting the burden required to establish a *prima facie* case of obviousness as to claims 14 and 32. Reversal of the Examiner's rejections of claims 14 and 32 is therefore respectfully requested.

F.      <u>Claims 12, 15-17, 29 and 31 are non-obvious over *Sedlar, Cooper* and *Wang*</u>.


<u>Dependent Claims 12 and 29</u>

Claim 12 depends from claim 1 and additionally recites the steps of: detecting a condition for pausing the converting step, and in response to the condition, pausing the converting step. Claim 29 depends from claim 26 and recites similar subject matter. In rejecting this claim, the Examiner adds *Wang* to *Sedlar* and *Cooper*. *Wang*, however, merely discloses a method of diagnosing system crashes, with the cited passages merely disclosing detecting a crash and restarting a system. Once again, the rejections are superficial and conclusory in nature, as there is no discussion of any reason why the cited passages in *Wang* would motivate one of ordinary skill in the art to pause or stop a filesystem conversion process. The Examiner has again fallen far short of meeting the burden required to establish a *prima facie* case of obviousness as to claims 12 and 29. Reversal of the Examiner's rejections is therefore respectfully requested.

Dependent Claims 15 and 31

Claim 15 depends from claim 1 and additionally recites detecting a condition for stopping the converting step, and in response to the condition, stopping the converting step. Claim 31 depends from claim 26 and recites similar subject matter. The Examiner cites the same passages in *Wang* for allegedly teaching these steps as were cited in connection with claims 12 and 29. Also, as with the rejection for claims 12 and 29, the rejections are superficial and conclusory in nature, as there is no discussion of any reason why the cited passages in *Wang* would motivate one of ordinary skill in the art to pause or stop a filesystem conversion process. The Examiner has again fallen far short of meeting the burden required to establish a *prima facie* case of obviousness as to claims 15 and 31. Reversal of the Examiner's rejections is therefore respectfully requested.

Dependent Claims 16 and 17

Claims 16 and 17 are not argued separately.

G.    Claim 13 is non-obvious over *Sedlar, Cooper, Wang* and *Official Notice*.

Claim 13 depends from claim 12, and additionally recites that the condition that triggers the pause is one of a product install on the filesystem; a restore operation involving the filesystem; and a back-up operation involving the filesystem. In rejecting this claim, the Examiner adds *Official Notice* to *Sedlar, Cooper* and *Wang*. The Examiner simply disregards the fact that none of these references discloses the recited conditions for pausing a conversion process with a conclusory allegation that it would have been obvious to pause based upon these conditions. Such a conclusory observation, however, absent any supporting evidence, falls far short of raising to the level necessary to sustain a *prima facie* case of obviousness. In fact, the passages cited in *Wang* in connection with the rejection of claim 12 focus on stopping, rather than pausing, a system, and the conditions that trigger stops are far different than the recited conditions that would trigger a pause in the conversion process recited in claim 13. The Examiner's analysis of claim 13 is once again superficial and conclusory in nature, and

insufficient to sustain an obviousness rejection. Reversal of the Examiner's rejection of claim 13 is therefore respectfully requested.

H.  Claim 27 is non-obvious over *Sedlar, Cooper* and *Dubinski*.

Claim 27 depends from claim 26 and additionally recites the program code is further configured to non-recursively build a list of directories of the first type. In rejecting this claim, the Examiner adds *Dubinski* to *Sedlar* and *Cooper*, but as with many of the other rejections, the cited passages in *Dubinski* are cited without any supporting analysis (beyond a conclusory statement) suggesting a reason why the teachings of *Dubinski* would motivate one of ordinary skill in the art to arrive at Applicants' claimed invention. The Examiner has not made a *prima facie* case of obviousness, and the rejection of claim 27 should be reversed.

I.  Claims 33 and 35 are non-obvious over *Sedlar, Cooper* and *Durand*.

Claim 33 depends from claim 32 and additionally recites that the program code is further configured to convert another directory, different than the particular directory, while the particular directory is being used more than the predetermined amount. Claim 35 depends from claim 26, and recites that the program code is further configured to execute at an adjustable priority level. In rejecting these claims, the Examiner adds *Durand* to *Sedlar* and *Cooper*, but as with many of the other rejections, the cited passages in *Durand* are cited without any supporting analysis (beyond a conclusory statement) suggesting a reason why the teachings of *Durand* would motivate one of ordinary skill in the art to arrive at Applicants' claimed invention. The Examiner has not made a *prima facie* case of obviousness, and the rejections of claims 33 and 35 should therefore be reversed.

## CONCLUSION

In conclusion, Applicants respectfully request that the Board reverse the Examiner's rejections of claims 1-35, and that the Application be passed to issue. If there are any questions

regarding the foregoing, please contact the undersigned at 513/241-2324.  Moreover, if any other charges or credits are necessary to complete this communication, please apply them to Deposit Account 23-3000.

<div style="text-align:center">

Respectfully submitted,

WOOD, HERRON & EVANS, L.L.P.

</div>

Date:   July 3, 2007

By:   /Scott A. Stinebruner/
      Scott A. Stinebruner
      Reg. No. 38,323

2700 Carew Tower
441 Vine Street
Cincinnati, Ohio 45202
(513) 241-2324

## VIII. CLAIMS APPENDIX:  CLAIMS ON APPEAL (S/N 10/777,869)

1.  (Original) A method for converting a filesystem from a first type to a second type, the method comprising the steps of:

> generating a list of directories of the first type in the filesystem to convert; and

> converting each directory in the list to the second type while maintaining the file system in an active state.

2.  (Original) The method of claim 1, further comprising the step of:

> sequentially initiating the steps of generating and converting upon initial program load of a computer system utilizing the filesystem.

3.  (Original) The method of claim 1, wherein the step of converting further includes the steps of:

> retrieving an identifier of a directory in the list;

> converting the directory to a second-type directory; and

> activating the second-type directory.

4.  (Original) The method of claim 3, further comprising the step of:

> repeating the steps of claim 3 for each directory in the list.

5.  (Original) The method according to claim 1 wherein the list represents a top-down view of the filesystem spanning from a root directory down to an outermost leaf-node.

6.  (Original) The method according to claim 5, wherein the step of converting each directory is performed for each directory in an order opposite to that of the list.

7.  (Original) The method according to claim 1, wherein the step of generating further includes the steps of:

> a)  adding a root directory as a current entry in the list;

> b)  identifying a child directory of the current entry in the list;

c) appending the identified child directory to the list;

d) repeating steps b) and c) for each child directory within the current entry;

e) changing a next directory in the list immediately following the current entry to be the current entry, if the next directory exists in the list; and

f) repeating steps b) - e) until no next directory exists in the list.

8. (Original) The method of claim 1, further comprising the step of:

marking a particular directory as being in the process of conversion once the particular directory is in the list.

9. (Original) The method of claim 8, wherein a new object added to the particular directory is appended at an end of the particular directory.

10. (Previously Presented) The method of claim 1, wherein the step of converting further includes the steps of:

a) creating a second-type root directory;

b) creating a second-type directory corresponding to a particular directory in the list;

c) generating a respective link in the second-type directory for each child object of the particular directory;

d) activating the second-type directory; and

e) removing the particular directory from the list.

11. (Original) The method of claim 10, further comprising the step of:

creating a data structure associated with the second-type directory, the data structure including a first anchor point that is associated with a parent directory of the directory and a second anchor point associated with a parent directory of the second-type directory.

12. (Original) The method of claim 1, further comprising the steps of:

    detecting a condition for pausing the converting step; and

    in response to the condition, pausing the converting step.

13. (Original) The method of claim 12, wherein the condition is one of:

    a product install on the filesystem; a restore operation involving the filesystem; and a back-up operation involving the filesystem.

14. (Original) The method of claim 1, further comprising the steps of:

    determining a usage rate of a particular directory before converting that directory; and

    postponing converting the particular directory based on the usage rate.

15. (Original) The method of claim 1, further comprising the steps of:

    detecting a condition for stopping the converting step; and

    in response to the condition, stopping the converting step.

16. (Original) The method of claim 15, wherein the condition is one of:

    a system crash, encountering a corrupted object within the filesystem, and insufficient available storage.

17. (Original) The method of claim 15, wherein the converting step is restarted upon a subsequent initial program load involving the filesystem.

18. (Original) The method of claim 3, wherein the step of activating further includes the steps of:

    identifying a data structure associated with the directory;

    changing the data structure to be associated with the second-type directory; and

    removing the directory.

19. (Original) The method of claim 18, further comprising the step of:

　　asserting a lock on first data structure while performing the step of changing.

20. (Original) A method for converting a filesystem from a first type to a second type, the method comprising the steps of:

　　executing a conversion process to convert each directory of the first type in the filesystem into the second type while maintaining the filesystem in an active state; and

　　terminating the conversion process when every directory of the first type in the filesystem has been converted to the second type.

21. (Original) The method of claim 20, further comprising the step of:

　　initiating the executing step upon initial program load of a computer system utilizing the filesystem.

22. (Original) The method of claim 20, further comprising the step of:

generating a list of directories of the first type in the filesystem to convert.

23. (Original) The method of claim 22, wherein converting each directory includes the steps of:

　　retrieving an identifier of a directory in the list;

　　converting the directory to a second-type directory; and

activating the second-type directory.

24. (Original) A program product, comprising:

　　a program code configured upon execution to:

　　　　generate a list of directories of a first type in a filesystem to convert; and

　　　　convert each directory in the list to a second type while maintaining the filesystem in an active state; and

　　a signal bearing medium bearing the program code.

25. (Original) A program product, comprising:

a program code configured upon execution to:

initiate a conversion process to convert each directory of a first type in a filesystem into a second type while maintaining the filesystem in an active state; and

terminate the conversion process when every directory of the first type in the filesystem has been converted to the second type; and

a signal bearing medium bearing the program code.

26. (Original) An apparatus comprising:

at least one processor;

a memory coupled with the at least one processor; and

a program code residing in memory and executed by the at least one processor, the program code configured to:

initiate a conversion process to convert each directory of a first type in a filesystem into a second type while maintaining the filesystem in an active state; and

terminate the conversion process when every directory of the first type in the filesystem has been converted to the second type.

27. (Original) The apparatus of claim 26, wherein the program code is further configured to:

non-recursively build a list of directories of the first type.

28. (Original) The apparatus of claim 26, wherein the program code is further configured to:

modify a particular directory being converted such that any new entries in that particular directory are appended at an end of that particular directory.

29. (Original) The apparatus of claim 26, wherein the program code is further configured to:

> pause the conversion process if a predetermined operation is detected as being performed on the filesystem.

30. (Original) The apparatus of claim 26, wherein the program code is further configured to:

> for a particular directory being converted, maintain a first version having the first type and a second version having the second type; and
>
> activate the second version while de-activating the first version.

31. (Original) The apparatus of claim 26, wherein the program code is further configured to:

> stop the conversion process at a particular point when an abnormal system condition is detected; and
>
> restart the conversion process at the particular point.

32. (Original) The apparatus of claim 26, wherein the program code is further configured to:

> for a particular directory being converted, determine if the particular directory is being used more than a predetermined amount; and
>
> postpone converting that particular directory while it is being used more than the predetermined amount.

33. (Original) The apparatus of claim 32, wherein the program code is further configured to:

> convert another directory, different than the particular directory, while the particular directory is being used more than the predetermined amount.

34. (Original) The apparatus of claim 26, wherein the program code is further configured to:

for a particular directory already converted, convert the particular directory back to the first type.

35. (Original) The apparatus of claim 26, wherein the program code is further configured to:

execute at an adjustable priority level.

## IX. EVIDENCE APPENDIX

None.

## X. RELATED PROCEEDINGS APPENDIX

None.